Esercitazione di ricapitolazione – parte II

Potete trovare molti esempi di programmi o spezzoni di programma Assembly e Macchine a Stati Finiti nei temi di esame passati. Altri ne potete trovare nelle slide e nel materiale delle esercitazioni. Seguono alcuni esercizi di ricapitolazione.

1. Sintetizzare la macchina a stati finiti che gestisce un distributore automatico di bibite. La macchina deve funzionare in modo che quando vengono inserite monete per esattamente 30 cents la macchina eroga una bottiglia di acqua. All'istante successiva la macchina si resetta qualunque sia l'input. Le monete che si possono inserire sono: 0, 10 centesimi. [7].

2. Tradurre in linguaggio macchina, le seguenti istruzioni assembly: [3]

```
L1:   lw $t0, 8($t1)        8:
      beq $t0, $zero, L1:   24:          …….
      j L1:                 32:
```

Facendo riferimento a: lw 0x26 rs rt offset, beq 0x4 rs rt offset, j 0x2 label, Ee ricordando che nel processore MIPS i registri: $zero, $t0, $t1 corrispondono rispettivamente ai registri $0, $8, $9.

3. Definire che cos'è una ISA. [1].

4. Tradurre in linguaggio assembly e poi in linguaggio macchina il seguente spezzone di codice C:

```
for (i=0; i<N; i++)
{     s = vett[i];
      if (s == 5)break;
} [5].
```

5. Disegnare lo schema generale di una macchina a stati finiti. Definire i passi per la progettazione e sintesi di una FSM [3].

6. Definire i formati delle istruzioni e darne un esempio [4].

7. Definire i tipi di istruzioni messi a disposizione dall'ISA del MIPS R3000 [2].

8. Cos'è il "Global pointer"? Perchè si utilizza? In che relazione è con i segmenti dati, testo e stack?

9. In che relazione sono i registri $sp e $fp con la memoria principale?

10. Cosa succede in un'operazione di push o di pop? Quale parte dell'architettura interessa? [1]

11. Modalità di accesso ai dati nella memoria principale (RAM) nelle architetture MIPS [2].

12. Sintetizzare una macchina a stati finiti in grado di:
- Leggere un carattere alfabetico (A, B, C o D).
- Concatenarlo agli altri caratteri letti in sequenza.
- Riconoscere la sequenza di caratteri: AAA.

13. Date le seguenti 2 procedure definite come sotto, determinare il contenuto del file eseguibile.
Procedura A. E' costituita da 128 istruzioni. I suoi dati statici ammontano ad 1kbyte. Queste sono le istruzioni iniziali della procedura:

```
add $t0, $t1, $t2
```

```
L1:  lw $t1, 16($gp)
     beq $t1, $t2, L1
     jal B
     lw $t6, 20($gp)
….
```

Procedura B. E' costituita da 100 istruzioni. I suoi dati statici ammontano ad 10kbyte. Queste sono le istruzioni iniziali della procedura:

```
L1:    add $t0, $t1, $t2
       jal B
       beq $t0, $t2, L3
       j L1:
L3:    sub $t1, $t2,$t3
```

14. Tradurre la procedura B definita sopra in linguaggio ad alto livello (C o Java).

15. Qual è lo spazio indirizzabile della memoria da un'istruzione di load/store?

16. Quali sono i passi necessari per passare da un programma ad alto livello ed un eseguibile binario? Qual è la funzione del loader? Cos'è la tabella dei simboli? Cosa si intende per rilocazione?

17. Tradurre in linguaggio Assembly mediante Jump Address Table il seguente spezzone di codice C:

```
switch (s3)
{      case 0: t2 = t1 + t0; s2 = s1 + s0; break;
       case 1: s2 = s1 + s0; break;
       default: break;
}
```

18. Spiegare cosa rappresenta una Jump Address Table ed i vantaggi nel suo utilizzo.

19. Cos'è il record di attivazione e de attivazione?

20. Cosa contengono i registri $t ed i registri $s? Quali sono le differenze di utilizzo?

21. Quali registri sono coinvolti nel passaggio dei parametri e che ruolo hanno?

22. Cosa si intende per salto indiretto? Quali istruzioni e registri sono coinvolti?